Calcolatori Elettronici L-A

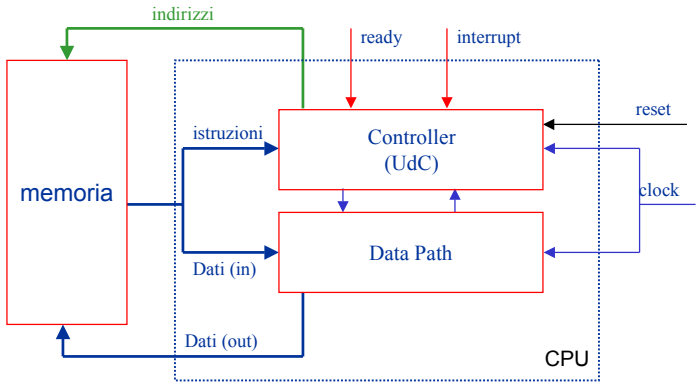
ISA DLX: Implementazione Tramite Struttura Sequenziale

Richiamo: caratterirtiche della CPU “DLX”

- “DLX” è un nome di fantasia (sta per De LuXe). Questa CPU è stata progettata a scopo didattico da John Hennessy e da David Patterson (ne è stata realizzata una molto simile a questa a Stanford Univ., il “MIPS”)
- Contiene molte delle soluzioni progettuali adottate nelle CPU commerciali di oggi ma (essendo teorica) non obbliga all’approfondimento di complicati dettagli tecnici, inutili dal punto di vista didattico
- Load-Store cioè carica gli operandi dalla memoria ai registri ➔ esegue le operazioni ➔ salva nuovamente in memoria il risultato contenuto nel registro. E’ una macchina R-R le cui istruzioni assembler hanno formato (3-operandi : 0-indirizzi di memoria)
- Progettata con filosofia RISC: poche e semplici istruzioni nel LM per favorire l’implementazione della pipeline
- Spazio di I/O memory-mapped
- Assenza di istruzioni assembler per la gestione dello Stack
- contiene registri da 32bit di uso generale (intercambiabili nella sintassi del LM = ortogonali)
- Sia il Bus Dati che quello Indirizzi hanno 32 linee (4GB indirizzabili)
- 1 word = 32 bit

Datapath e Unità di Controllo (1)

- La struttura di una CPU, come tutte le reti logiche sincrone che elaborano dati, può essere strutturata in due blocchi Controller (Unità di Controllo) e Datapath.
- La CPU, per funzionare, ha bisogno della memoria esterna su cui risiedono il programma e i dati.



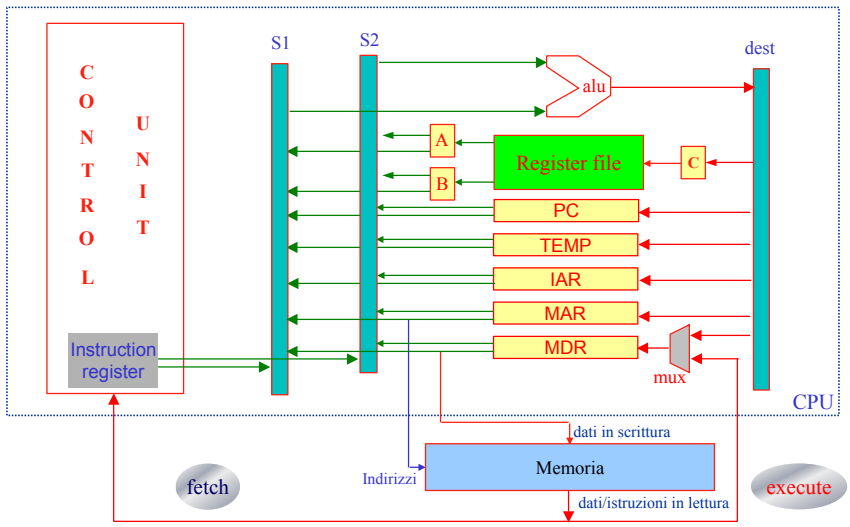
Datapath e Unità di Controllo (2)

• DATAPATH: contiene tutte le unità di elaborazione ed i registri necessari per l'esecuzione delle istruzioni della CPU. Ogni istruzione appartenente all'Instruction Set è eseguita mediante una successione di *operazioni elementari,* dette *micro-operazioni*.

• *Micro-operazione*: operazione eseguita all'interno del DATAPATH *in un ciclo di clock* (esempi: trasferimento di un dato da un registro ad un altro registro, elaborazione ALU)

• CONTROLLER: è una RSS che in ogni ciclo di clock invia un ben preciso insieme di segnali di controllo al DATAPATH al fine di specificare l'esecuzione di una determinata *micro-operazione*.

Struttura del DLX (esecuzione sequenziale)



Parallelismo dell'architettura: 32 bit
(bus, alu e registri hanno parallelismo 32)

I segnali di controllo non sono indicati!

Struttura del DLX (esecuzione sequenziale)

- Instruction Register il registro deputato a contenere l'opcode dell'istruzione che deve essere eseguita.
- Bus S1 ed S2 portano gli operandi alla ALU. Sono multiplexer distribuiti con parallelismo 32bit
- Bus dest trasporta il risultato in uscita dalla ALU. In quanto bus essi non hanno la capacità di memorizzare questi dati. L'unica via di comunicazione tra S1-S2 e Dest passa attraverso l'ALU.
- Memory Data Register memorizza i dati per istruzioni Load/Store in ram prima che arrivino in ram (store) o nei registri (load)
- Memory Address Register contiene l'indirizzo di accesso alle celle di memoria oggetto delle istruzioni del caso precedente
- Register File banco dei 32 registri GPR da 32bit. Campionano sul fronte positivo del ck (ET), hanno un input WE# e due input OE# perché le loro due uscite possono andare su A che su B...
- A, B e C sono tre buffer del banco di registri RF: A, B (di lettura dai regs) e C (di scrittura sui regs). Ad ogni ciclo di clock (sull'edge) è possibile accedere in lettura ad una sola coppia di registri interni campionandone il contenuto su A, B. Allo stesso modo in un Tck è possibile scrivere su un solo registro il contenuto di C.
- Program Counter contiene sempre l'indirizzo in memoria della prossima istruzione da eseguire calcolato durante il fetch sommando 4 all'indirizzo di provenienza dell'istruzione appena prelevata.
- Interrupt Address Register contiene "l'indirizzo di ritorno" cioè il contenuto del Pc salvato al momento di trasferire il controllo all'indirizzo della procedura di servizio all'interrupt
- Temporary register registro temporaneo

Tempo necessario ad un trasferimento dati sul Datapath

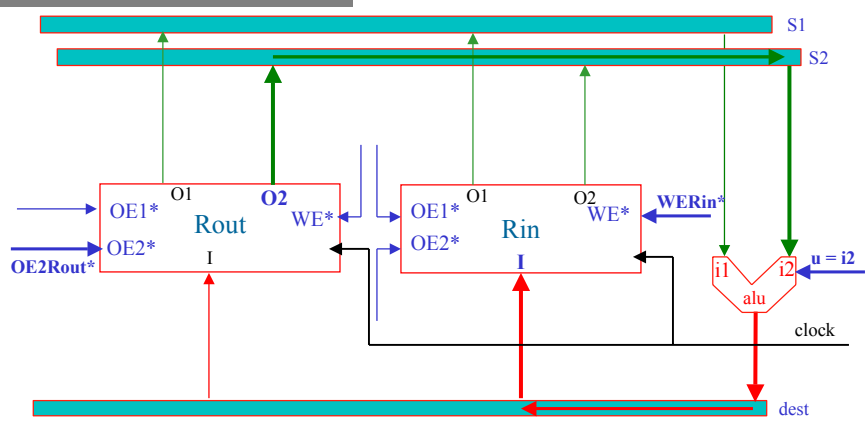
- Al fine di valutare la massima frequenza a cui è possibile far funzionare il datapath è importante conoscere le seguenti temporizzazioni:
  - $T_C\,(max)$: ritardo max tra il fronte positivo del clock e l'istante in cui i segnali di controllo generati dall'unità di controllo sono validi;
  - $T_{OE}\,(max)$: ritardo max tra l'arrivo del segnale OE e l'istante in cui i dati del registro sono disponibili sul bus;
  - $T_{ALU}\,(max)$: ritardo massimo introdotto dalla ALU;
  - $T_{SU}\,(min)$: tempo di *set-up* minimo dei registri (requisito minimo per il corretto campionamento da parte dei registri).
- La massima frequenza di funzionamento del datapath si calcola come segue:

$$T_{CK} > T_C\,(max) + T_{OE}\,(max) + T_{ALU}\,(max) + T_{SU}\,(min)$$

$$f_{CK}(max) = 1/T_{CK}$$

Esempio: esecuzione della microistruzione Rin ← Rout

I segnali in blu (*segnali di controllo*) provengono dall'Unità di Controllo



I segnali di controllo in grassetto sono attivi nel ciclo di clock in cui il micro-step Rin ← Rout viene eseguito

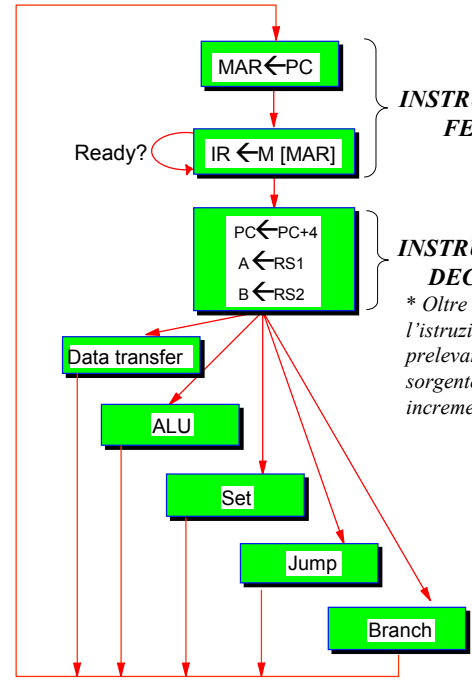
Esercizio

• **Si disegnino le forme d'onda dei segnali di controllo e dei dati per la microistruzione vista nel lucido precedente riferendole al clock e considerando i parametri temporali visti nel calcolo di f_{ck}(max).**

Il progetto dell'Unità di Controllo

• Una volta definito il Set di Istruzioni e progettato il DATAPATH, il passo successivo del progetto di una CPU è il progetto dell'Unità di Controllo (*CONTROLLER*).
•Il CONTROLLER è una RSS: il suo funzionamento può essere specificato tramite un *diagramma degli stati*.
•Il CONTROLLER (come tutte le RSS) permane in un determinato stato per un ciclo di clock e transita (*può transitare*) da uno stato all'altro in corrispondenza degli istanti di sincronismo (fronti del clock).
•Ad ogni stato corrisponde quindi un ciclo di clock. Le *micro-operazioni* che devono essere eseguite in quel ciclo di clock sono specificate (in linguaggio RTL) nel diagramma degli stati che descrive il funzionamento del CONTROLLER *all'interno degli stati*.
•A partire dalla descrizione data in RTL si sintetizzano poi i segnali di controllo che devono essere inviati al DATAPATH per eseguire le operazioni elementari associate ad ogni stato.

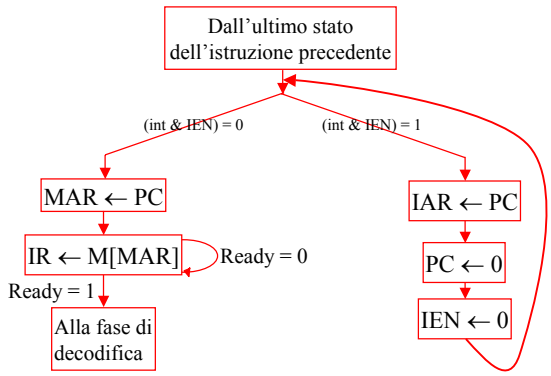
Il diagramma degli stati del controller (1)



Quanti clock sono necessari per eseguire la fase di fetch?

Fase di Fetch e gestione interrupt

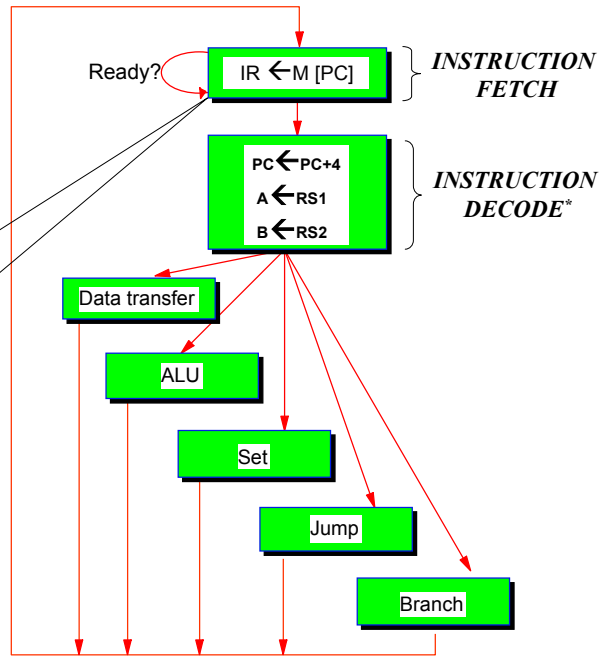
- In questa fase si deve verificare se è presente un interrupt (evento esterno asincrono che la CPU deve "servire" con apposito software);
- se l'interrupt è presente e può essere servito (IEN = true, Interrupt Enable Flag) si esegue implicitamente l'istruzione di chiamata a procedura all'indirizzo 0, e si salva l'indirizzo di ritorno nell'apposito registro IAR (Interrupt Address Register);
- se l'interrupt non è presente e le interruzioni non sono abilitate, si va a leggere in memoria la prossima istruzione da eseguire (il cui indirizzo è in PC)



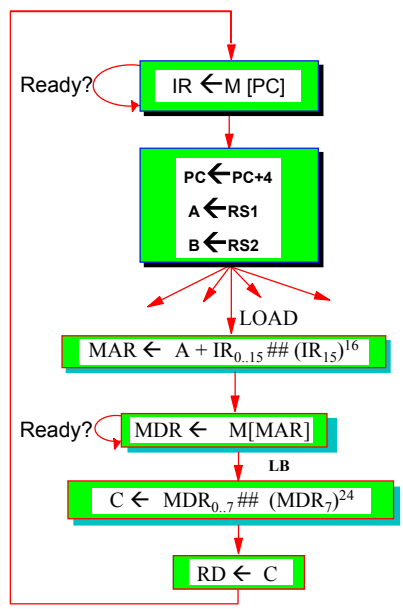
Il diagramma degli stati del controller (2)

Si modifica il DATAPATH in maniera da poter indirizzare la memoria da PC. Tutte le istruzioni impiegano un clock in meno per essere eseguite !

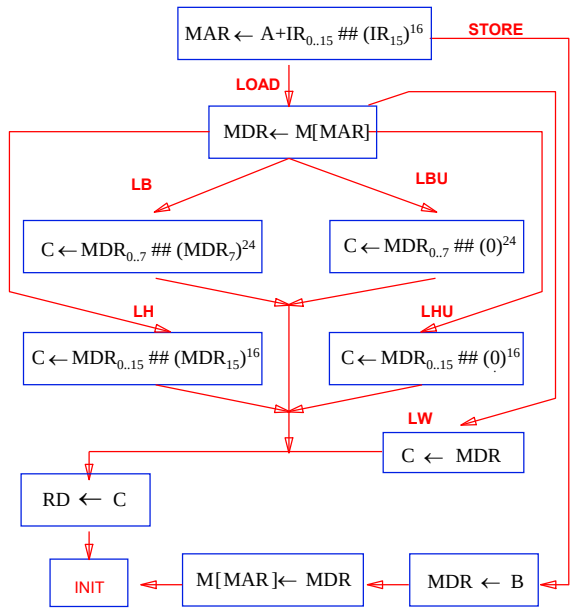
Quanti clock sono necessari per eseguire la fase di fetch?



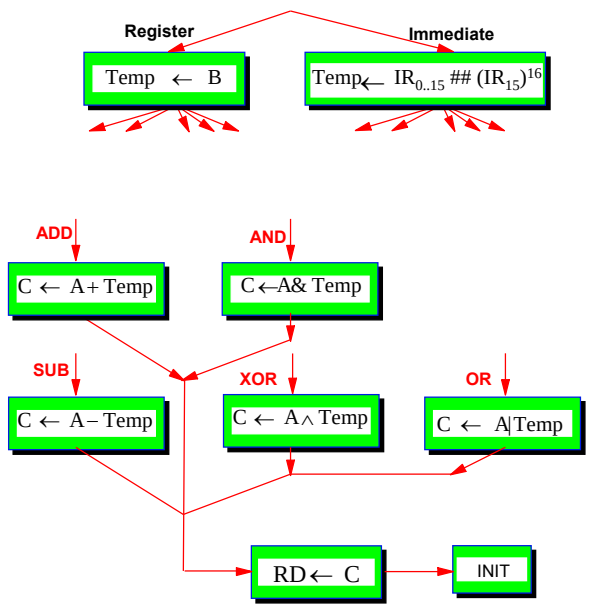
Controllo per l'istruzione LB (Load Byte)



Controllo per le istruzioni di Data Transfer

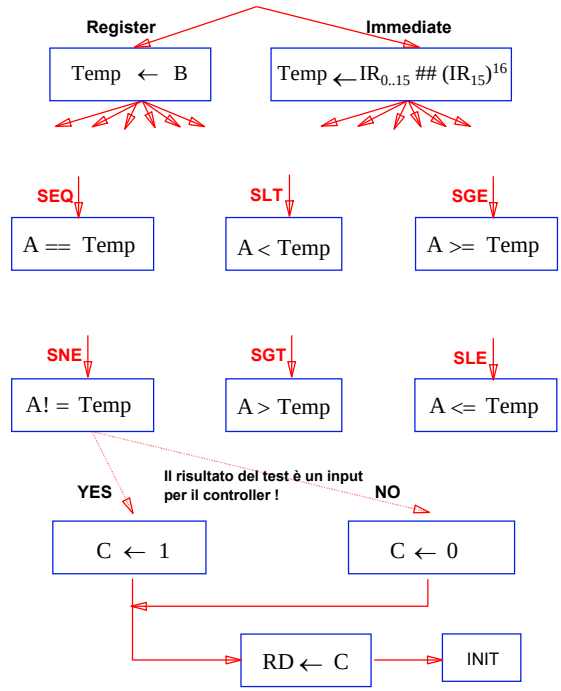


Controllo per le istruzioni ALU

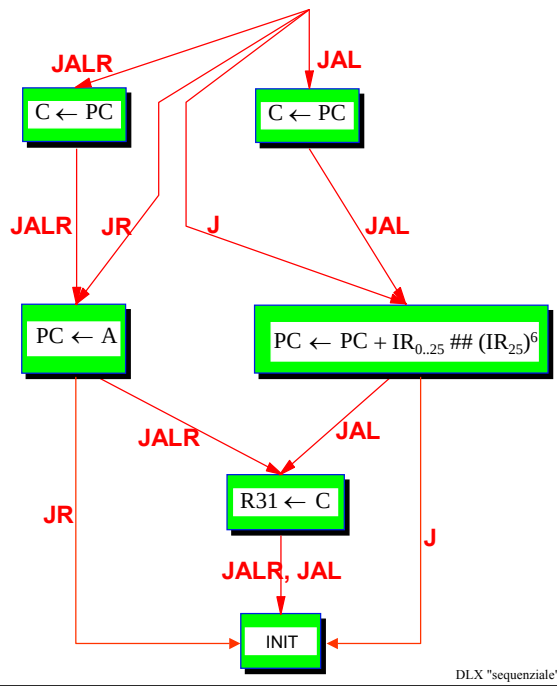


Controllo per le istruzioni di SET (confronto)

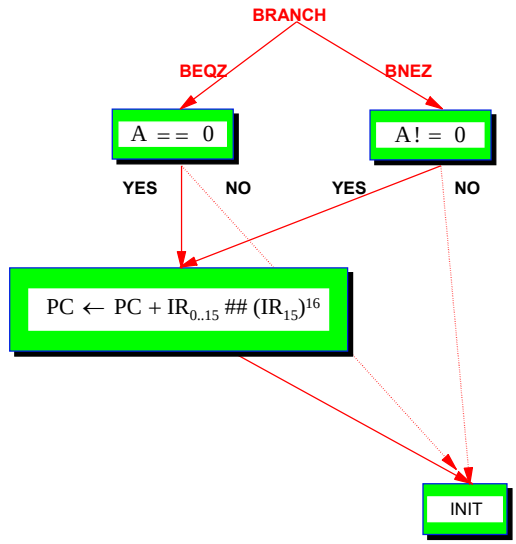
SET *condition* IN REGISTER:
S*cond* Rd, Rs1, Rs2
(dove *cond*: EQ, LT, NEQ, …)



Controllo per le istruzioni di Salto



Controllo per le istruzioni di Branch



Numero di clock necessari per eseguire le istruzioni

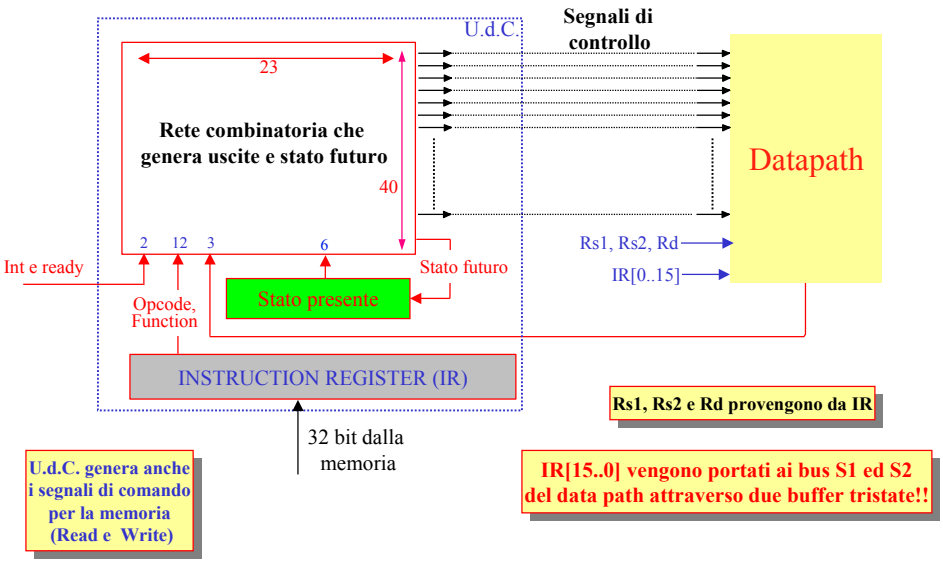
Istruzione	Cicli	Wait	Totale
Load	6	2	8
Store	5	2	7
ALU	5	1	6
Set	6	1	7
Jump	3	1	4
Jump and link	5	1	6
Branch (taken)	4	1	5
Branch (not taken)	3	1	4

$$CPI = \sum_{i=1}^{n} \left(CPI_i * \frac{N_i}{\text{numero totale di istruzioni}} \right)$$

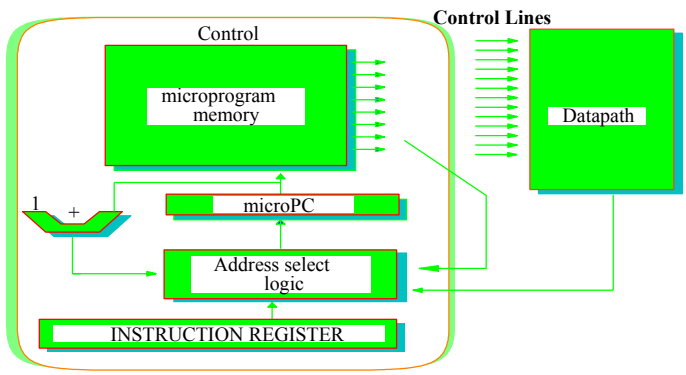
Esempio: GCC su DLX
LOAD: 21%, STORE: 12%, ALU: 37%, SET: 6%, JUMP: 2%, BRANCH (taken): 12%, BRANCH (not-taken): 11%

⟹ CPI = 6.3

Controllo cablato ("*hardwired*")



Unità di controllo con architettura microprogrammata



μ - INSTRUCTION

Esercizi

- Quali segnali di controllo devono essere attivi per eseguire le micro-operazioni di prelievo degli operandi sorgente dal Register File (A ← Rs1 , B ← Rs2) ?
- Si scrivano le micro-operazioni necessarie ad eseguire l'istruzione di ritorno dalla procedura di servizio dell'interrupt (detta anche Return from Exception o RFE)
- Si consideri la sequenza di micro-operazioni necessarie per mettere in esecuzione la procedura di servizio dell'interrupt. Perché è necessario eseguire la micro-operazione: IEN ← 0?; cosa succederebbe senza questa micro-operazione ?
- Qual è l'istruzione assembler del DLX che si deve eseguire per tornare da una procedura al programma chiamante? (si ricordi come viene eseguita l'istruzione JAL)

I passi dell'esecuzione delle istruzioni

Nel DLX l'esecuzione di tutte le istruzioni può essere scomposta in *5 passi*, ciascuno eseguito in uno o più cicli di clock.

Tali passi sono detti:

1) FETCH: l'istruzione viene prelevata dalla memoria e posta in IR.

2) DECODE: l'istruzione in IR viene decodificata e vengono prelevati gli operandi sorgente dal Register File.

3) EXECUTE: elaborazione aritmetica o logica mediante la ALU.

4) MEMORY: accesso alla memoria e, nel caso di BRANCH aggiornamento del PC ("branch completion").

5) WRITE-BACK: scrittura sul Register File (registro del banco).

Le *micro-operazioni* eseguite in ciascun passo (1)

1) FETCH

MAR ← PC ;
IR ← M[MAR];

2) DECODE

A← RS1, B← RS2, PC← PC+4

Le *micro-operazioni* eseguite in ciascun passo (2)

3) EXECUTE

- **Memoria:**

  MAR ← A + $IR_{0..15}$ ## $(IR_{15})^{16}$;

  MDR← B; ***(RD=RS2 nelle STORE)***

- **ALU:**

  C ← A *op* B (*oppure* $IR_{0..15}$ ## $(IR_{15})^{16}$) ;

- **Branch:**

  Temp ← PC + $IR_{0..15}$ ## $(IR_{15})^{16}$;

  Cond ← A *op* 0 ;

Le *micro-operazioni* eseguite in ciascun passo (3)

4) MEMORY

- **Memoria:**

  MDR ← M[MAR]; (*LOAD*)

  M[MAR] ← MDR; (*STORE*)

- **Branch:**

  If (Cond) PC ← Temp;

5) WRITE-BACK

C ← MDR; (*se è una LOAD*)

RD ← C ;

Esercizio tipo esame

Si supponga di voler estendere il Set di Istruzioni del DLX con la seguente istruzione:

ADD Rx, IMMEDIATE (Ry)

dove Rx, Ry sono due registri di uso generale e IMMEDIATE è un numero di 16 bit con segno.
L'istruzione deve eseguire la somma fra Rx e la word (32 bit) situata all'indirizzo di memoria Ry+IMMEDIATE e quindi deve scrivere il risultato in Rx.

a) Quale fra i 3 formati delle istruzioni DLX risulta il più idoneo alla codifica di questa nuova istruzione ? Come potrebbe essere codificata in binario l'istruzione ADD R1, 20 (R2)?
b) Con riferimento al datapath visto a lezione, si sviluppi il diagramma degli stati che controlla l'esecuzione dell'istruzione ADD Rx, IMMEDIATE (Ry), inserendo anche gli stati necessari alle fasi di fetch e decodifica.
c) A partire dal diagramma degli stati ottenuto e ipotizzando che ogni accesso alla memoria richieda 3 clock, si calcoli il CPI (clock-per-istruzione) della nuova istruzione.